IMPLEMENTASI *DOCKER* UNTUK PENGELOLAAN BANYAK APLIKASI *WEB* (Studi Kasus : Jurusan Teknik Informatika UNESA)

M. Fadlulloh Romadlon Bik
D3 Manajemen Informatika, Fakultas Teknik, Universitas Negeri Surabaya, bickcorp95@gmail.com

Asmunin
Jurusan Teknik Informatika, Fakultas Teknik, Universitas Negeri Surabaya, asmunin@yahoo.com

Abstrak

Sistem *web hosting modern*, di dalam setiap servernya, mengelola banyak aplikasi *web*. Teknologi virtual machine dimanfaatkan untuk menyelesaikan masalah heterogenitas (perbedaan versi *library* atau tool dari beberapa aplikasi web). Peningkatan jumlah aplikasi web yang harus dihosting harus diikuti dengan peningkatan kualitas ataupun kuantitas sumber daya, terlebih saat hadirnya kebutuhan high availability dari layanan *web* tersebut. Teknik kontainerisasi (virtualisasi berbasis container) hadir sebagai solusi dan menjadi trend saat ini. *Docker* adalah salah satu *software* yang mengadopsi teknik kontainerisasi dan semakin banyak diterapkan di dalam lingkungan *web hosting*. Tulisan ini mencoba untuk melakukan kajian literatur terhadap teknologi virtualisasi di atas, baik virtual machine maupun container dan kemudian merangkum perbandingannya. Arsitektur *container* di dalam *Docker* merupakan fokus dari tulisan ini, termasuk perkembangan dan keunggulan dari *Docker* yang sudah diteliti dan diimplementasikan dalam dua tahun terakhir. *Docker* memudahkan proses *deployment* (penyebaran) aplikasi web beserta software pendukung seperti *web server, database server,* dependensi dan *environment* lain ke server. Dapat kemudahan dengan proses deployment (penyebaran) aplikasi web beserta software pendukung seperti web server, database server, dependensi dan environment lain ke server. Memberikan solusi pada banyak Aplikasi web yang membutuhkan *Docker* untuk bereksperimen atau mendukung mahasiswa yang ingin melakukan tugas akhir dengan berbagai topik.

Kata Kunci : *Docker*, Aplikasi *web, Container*

Abstract

The modern web hosting system, within each of its servers, manages many web applications. Virtual machine technology is used to solve the problem of heterogeneity (different versions of libraries or tools of some web applications). Increasing the number of web applications that must be hosted must be followed by an increase in quality or quantity of resources, especially when the presence of high availability requirements of the web service. Containerization technique (container-based virtualization) comes as a solution and becomes the current trend. Docker is one software that adopts containerization techniques and is increasingly applied in the web hosting environment. This paper attempts to conduct a literature review of the above virtualization technologies, both virtual machines and containers and then summarizes the comparison. The container architecture inside the Docker is the focus of this paper, including the developments and advantages of Dockers that have been researched and implemented in the last two years. Docker facilitates the deployment of web applications along with supporting software such as web servers, database servers, dependencies and other environments to the server. Can ease with the deployment of web applications and supporting software such as web servers, database servers, dependencies and other environments to the server. Provide solutions on many Web applications that require Dockers to experiment or support students who want to do final projects on various topics.

Keywords*: Docker, Web app, Container.*

PENDAHULUAN

Perkembangan aplikasi berbasis *web* sangat pesat, seiring dengan perkembangan komputer dan *internet.* Selain itu, aplikasi berbasis *web* juga semakin banyak digunakan karena dapat diakses di berbagai platform komputer hanya dengan menjalankan *web browser*. Sehingga, kemudahan proses *deployment* (penyebaran) aplikasi *web* beserta software pendukung seperti *web server, database server,* dependensi dan *environment* lain ke *server* sangat dibutuhkan. Secara umum ada dua metode *deployment* aplikasi web kedalam *server*. Pertama menginstall *web* aplikasi beserta *environment* yang dibutuhkan ke dalam *server* tunggal, kelebihannya adalah *setup server* mudah, *simple* dan cepat dalam proses *deployment*. Tetapi metode tersebut memiliki kekurangan yaitu setiap aplikasi tidak tersiolasi, sehingga apabila mendeploy beberapa aplikasi yang masing-masing memiliki ketergantungan dengan paket versi tertentu dapat menimbulkan konflik dependensi (*dependecy hell*). Metode yang kedua yaitu dengan memanfaatkan teknologi virtualisasi berbasis *hypervisor*, jadi setiap aplikasi dan *dependency* yang dibutuhkan *dideploy* kedalam *Virtual Machine* (VM) yang berbeda. Dengan metode ini dapat meningkatkan scalabilitas,

karena setiap aplikasi berjalan pada *resource* (CPU, memory, I/O) yang berbeda sehingga dapat dengan mudah ditambahkan sesuai kebutuhan. Akan tetapi masalah klasik menjalankan *virtual machine* berbasis *hypervisor* adalah membutuhkan *resource* yang besar. Karena setiap VM menjalankan *guest OS* beserta kernelnya sendiri terpisah dari host. Sehingga ketika menjalankan aplikasi yang munkin besarnya hanya puluhan MB, VM juga harus menjalankan *guest OS* yang besarnya bisa mencapai 10GB. Maka dibutuhkan teknologi yang dapat menyediakan virtualisasi ringan (*Lightweight virtualization*) yang mengisolasi aplikasi beserta *environment* yang dibutuhkan dengan kebutuhan *resource* minimal yang dapat berjalan di berbagai infrastruktur untuk memudahkan proses *deployment* aplikasi.

Dari hal tersebut, Menjalankan virtual machine berbasis *hypervisor* adalah membutuhkan resource yang besar. Karena setiap VM menjalankan guest OS beserta kernelnya sendiri terpisah dari host. Sehingga ketika menjalankan aplikasi yang munkin besarnya hanya puluhan MB, VM juga harus menjalankan guest OS yang besarnya bisa mencapai 10GB. maksud penulis melaksanakan penelitian ini yaitu memudahkan proses *deployment* (penyebaran) aplikasi web beserta software pendukung seperti *web server*, *database server*, dll ke server. maka dapat mengambil kesimpulan permasalahan yang ada diantaranya adalah bagaimana mengimplementasikan *Docker* untuk pengelolaan banyak aplikasi web di jurusan teknik informatika Universitas Negeri Surabaya.

Berdasarkan masalah di atas, penulis memandang penting untuk melakukan penelitian yang selanjutnya dituangkan dalam bentuk Tugas Akhir dengan berjudul : ”Implementasi Docker Untuk Pengelolaan Banyak Aplikasi Web” (Studi Kasus : Jurusan Teknik Informatika Unesa)”.

KAJIAN PUSTAKA

DOCKER

Docker menurut sugianto, 2016 adalah suatu platform terbuka bagi pengembang perangkat lunak dan pengelola sistem jaringan untuk membangun, mengirimkan dan menjalanan aplikasi-aplikasi terdistribusi. Definisi tersebut membawa pengertian praktis bahwa *Docker* merupakan suatu cara memasukkan layanan ke dalam lingkungan terisolasi bernama container, sehingga layanan tersebut dapat dipaketkan menjadi satu bersama dengan semua pustaka dan software lain yang dibutuhkan.

Docker mempengaruhi pengembang sehingga yakin bahwa layanan tersebut akan berjalan dimanapun *Docker* berjalan. Ada dua masalah penting yang diselesaikan oleh *Docker*, yang pertama adalah beratnya dan besarnya sumber daya komputer yang digunakan oleh salinan OS yang berjalan di atas *hypervisor* yang berjalan di atas hardware fisik yang kedua, ungkapan “tadi aplikasi ini bekerja di komputer saya tetapi sekarang tidak bekerja seperti tadi”.

Dengan *Docker*, pengembang aplikasi bekerja dengan anggapan “apa yang dibangun dan dijalankan saat pengembangan dan test adalah sama dengan yang dibangun dan jalankan saat produksi”. Ini sejalan dengan *Miell & Sayers* yang menyatakan bahwa *Docker* adalah solusi *standard* untuk menyelesaikan salah satu area berbiaya tinggi dalam siklus pengembangan perangkat lunak, yaitu *deployment*. *Docker* memberikan beberapa keuntungan bagi pengembang perangkat lunak, termasuk dapat menggantikan peran dari *VM*, memudahkan pembuatan prototipe banyak *software*, dengan setiap software dan file terkait ada di *container* terisolasi menyederhanakan pemaketan *software* sesuai dengan kemampuan pengembang bukan mengikuti kemampuan *administrator web hosting* mengaplikasikan arsitektur *microservice* memodelkan jaringan (terutama data center) memungkinkan produktifitas *full-stack* ketika *offline* mengurangi biaya *debugging* memudahkan dokumentasi ketergantungan dan *touchpoints* dari *software* memungkin *delivery* berkelanjutan. Hal tersebut di disampaikan juga oleh Matthias dan Kane.

Boettiger melengkapinya dengan menyatakan bahwa *Docker* mampu melakukan virtualisasi pada *level* sistem operasi, *men-deploy container* secara portabel meskipun lintas *platform*, menyediakan fitur pemanfaatan ulang komponen, *sharing*, *archiving*, dan versioning dari image container. Teknologi container telah diimplementasikan oleh banyak penyedia layanan online di bidang *cloud computing* dengan pendekatan dan kelebihan/kekurangannya masing-masing.

Komponen *Docker*

a. *Docker images* merupakan *read-only* template untuk menjalankan containers. Sebuah *Image* dapat terdiri dari sistem operasi dan beberapa aplikasi yang sudah terinstall *Images* dapat ditumpuk berlapis dengan *images* lainnya, image yang paling atas disebut dengan *parent image* dan *image* yang paling bawah disebut dengan *base image*, Contohnya sebuah image yang berisi system operasi Ubuntu dengan Apache dan Aplikasi *Web* yang telah diistall *Image* ini digunakan untuk menjalankan *container*.
b. *Docker Container* adalah sebuah *directory* yang menyimpan segala sesuatu yang diperlukan agar aplikasi dapat berjalan, Setiap *container* dijalankan dari docker image yang telah ditentukan *container* dapat dijalankan *(run)*, diberhentikan *(stop)* dan dihapus *(remove) Container* merupakan platform yang menyedikan wadah yang terisolasi untuk aplikasi dan merupakan sebuah image bersifat *read-write* yang berjalan di atas image Docker menggunakan *union-file system* sebagai *back-end file system containernya*, dimana setiap perubahan yang disimpan pada *container* akan menyebabkan terbentuknya layer baru di atas *base image*, Jadi *container* merupakan layer dimana kita bisa melakukan instalasi aplikasi di dalamnya. Masing-masing *container* yang berjalan terisolasi dalam satu lingkungan dan *platform* aplikasi yang aman, tidak saling bentrok dengan aplikasi lain dalam *host* yang sama.
c. *Docker registry* adalah sebuah repositori (*publik* atau *private*) yang menyedikan ribuan *docker images*, Publik docker *registries* disebut dengan Docker hub,

User dapat melakukan perintah *push* melalui docker *client* ke docker *registry* untuk penyimpanan dan sharing. Dan pengguna lain dapat melakukan perintah *pull* untuk mendownload dan menjalakannya secara langsung.

d. *Docker File* merupakan sebuah skrip otomasi (builder) yang membangun sebuah *image*, Sebuah *Docker File* merupakan dokumen text atau skrip yang berisi semua perintah yang biasanya kita lakukan manual untuk membangun sebuah image, Dengan menggunakan perintah *docker build* dari terminal, kita akan melihat *Docker* membangun image secara bertahap berdasarkan eksekusi perintah dalam script.

e. *Docker* menggunakan kata mirip dengan yang digunakan pada Github dan *source control system* lainnya, namun jenis yang berbeda, *Repository* berupa ID untuk setiap image yang disimpan dalam *registry*, Ketika menjalankan perintah *docker commit* maka image itu akan kita beri nama denga *format username/name image*. Ketika kta mengupload image tersebut dengan perintah docker *push*, *index* akan melihat nama image dan memastikan tidak ada nama repository yang sama, jika tidak maka index akan memeriksa apakah memiliki akses terhadap *repository* tersebut, maka selanjutnya diijinkan untuk mengupload *image* versi baru ke *repository* tersebut.

f. *Docker index* terkait dengan *Hub Registry*, meski keduanya memiliki fungsi yang berlainan, *Index* mengatur *user account*, *permission*, *search*, tagging dan hal lain yang tersimpan pada *web interface public*. Ketika melakukan eksekusi perintah docker run untuk menjalankan suatu docker *image*, hal itu digunakan untuk mencari data pada *index* bukan *registry*. Ketika menjalankan perintah *docker pull* ataupun docker *push*, index akan menentukan apakah diijinkan untuk mengakses atau memodifikasi *image*, dan selanjutnya *registry* adalah bagian yang akan menyimpan *image* tersebut setelah mendapatkan hak akses dari *index*.

METODE

Analisa Sistem

Analisis sistem dilakukan dengan cara menguraikan suatu sistem informasi yang utuh ke dalam bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan yang akan ditemukan kelemahan dan kelebihan pada sistem tersebut.

Setiap aplikasi *web* beserta *environment* yang dibutuhkan di deploy kedalam containers. Setiap *containers*akan memiliki IP *Private* yang hanya dapat diakses oleh *Host*. Agar dapat diakses dari luar setiap *containers*akan dibuatkan domain. Dan untuk mengarahkan domain ke *container* tujuan digunakan *Nginx* sebagai *reverse proxy*.

Gambaran Sistem

Setiap aplikasi *web* beserta *environment* yang dibutuhkan di *deploy* kedalam *containers*. Setiap containers akan memiliki IP *Private* yang hanya dapat diakses oleh *Host*. Agar dapat diakses dari luar setiap containers akan dibuatkan domain. Dan untuk mengarahkan domain ke container tujuan digunakan Apache sebagai *reverse proxy*.

Alat Dan Bahan

Tabel 1. Spesifikasi Server

NO	Hardware	Spesifikasi
1	RAM	3 GB
2	Processor	Intel Core Duo
3	Harddisk	320 GB
4	Sistem Operasi	Windows 10
5	IP Publik	192.168.1.126

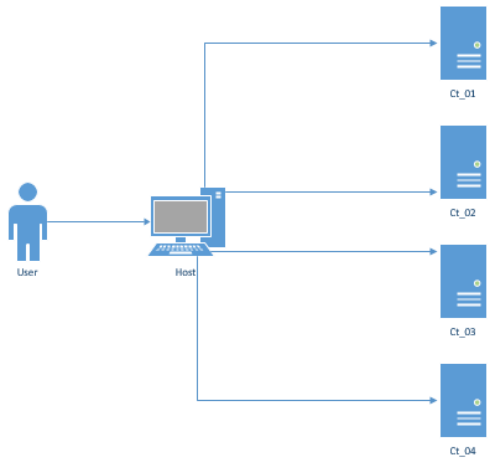
Tabel 2. Spesifikasi Klien

NO	Hardware	Spesifikasi
1	RAM	2 GB
2	Processor	Core I3
3	Harddisk	500 GB
4	Sistem Operasi	Windows 7

Desain Sistem

Desain Proses

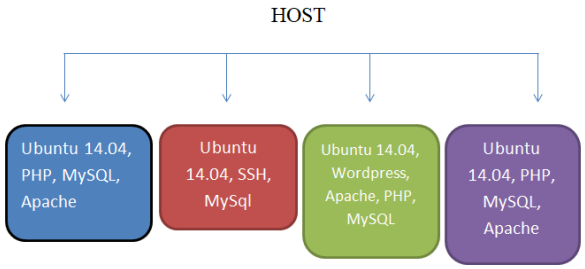
Docker pada tugas akhir ini dilakukan pada penyedia platform terbuka dalam bentuk teknologi virtualisasi berbasis *container*, ditujukan bagi para developer sysadmin untuk dapat membangun, membundel dan menjalankan aplikasi dimanapun dalam satu container yang ringan. Mirip seperti *Virtual Machine* (VM) namun lebih ringan karena docker tidak membawa keseluruhan sistem operasi, melainkan berbagi sistem dengan host induknya. Berikut ini desain sistem yang akan dibuat.



Gambar 1. Desain Sistem

Menentukan *Containers* Yang Akan Di Jalankan

Setelah perancangan *system* dibuat selanjutnya adalah menentukan aplikasi apa saja yang akan dijalankan pada setiap *container*. Dibawah ini *container* yang akan dijalankan dan daftar aplikasi yang terinstall pada masing-masing *containers*.

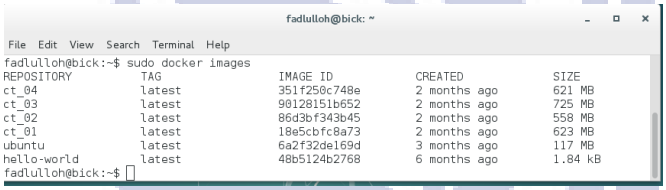


Gambar 2. *Containers* yang di jalankan

Implementasi Sistem

Build Docker images

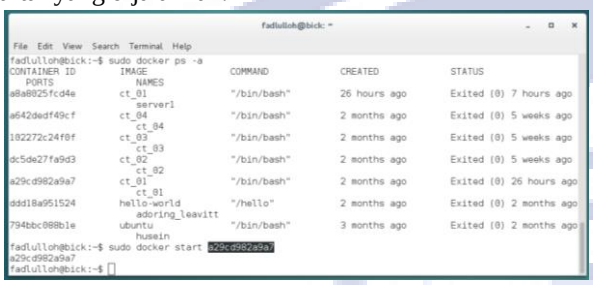
Images digunakan untuk menjalankan containers, maka dari itu pada tahapan ini penulis akan membangun (*build*) semua docker image yang dibutuhkan untuk menjalankan containers.



Gambar 3. List *Docker Images*

Menjalankan *Containers*

Setelah mem *build* semua images yang dibutuhkan selanjutnya adalah menjalankan *containers* menggunakan images tersebut. Berikut ini daftar yang dijalankan.



Gambar 4. List *Docker Containers*

Konfigurasi Apache sebagai Proxy

Setelah semua container dijalankan, selanjutnya adalah memberikan domain untuk setiap *containers.* Pada penelitian kali ini, penuls akan menggunakan domain IP. Berikut tabel nama *containers* dan domain yang akan digunakan.

Tabel 3. List Nama *Container* dan *Domain*

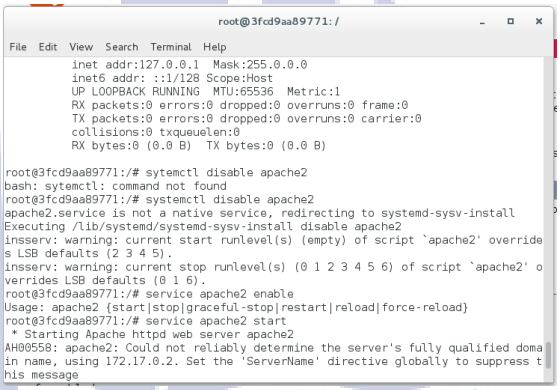
NO	Nama *Containers*	*Domain*
1	Ct_01	172.17.0.2
2	Ct_02	172.17.0.3
3	Ct_03	172.17.0.4
4	Ct_04	172.17.0.5

Ssetelah itu kemudian install Apache pada host dan mengkonfigurasinya sebagai proxy. Apache tersebut berfungsi untuk menerima HTTP *request* kemudian mengarahkannya ke *containers* sesuai dengan domain yang telah ditentukan.

HASIL DAN PEMBAHASAN

1. *Web server Containers*

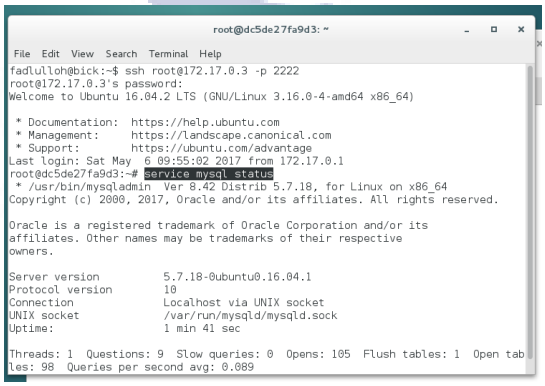
Halaman dibawah ini merupakan halaman untuk proses akses domain ke setiap *containers* dengan melalui *web browser* dari *client.*



Gambar 5. ***Web server Containers***

2. SSH *Container*

Halaman di bawah ini merupakan halaman untuk memastikan bahwa SSH *server* berjalan dengan baik maka akan dilakukan pengujian tersebut.

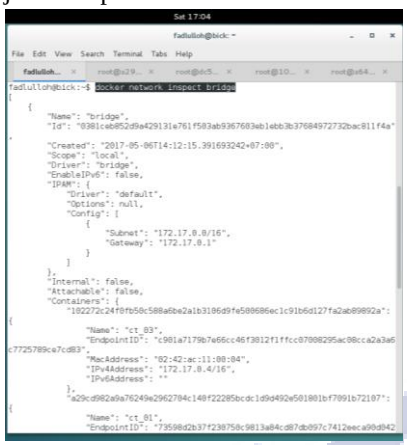


Gambar 6. SSH *Container*

3. Info *Container*

Pada halaman dibawah ini merupakan untuk mengetahui informasi lengkap pada *containers*,

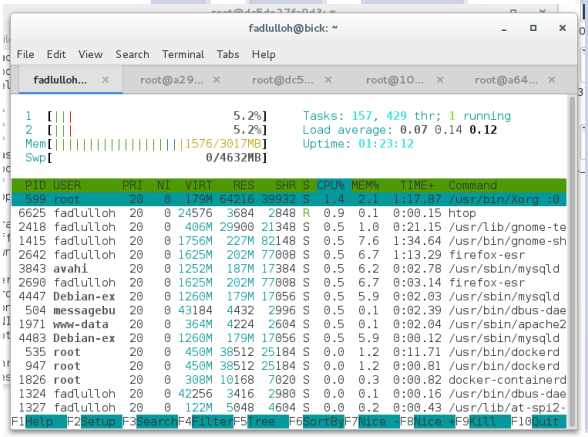
dapat menggunakan command *docker inspect* dan menjalankan perintah *docker* info.



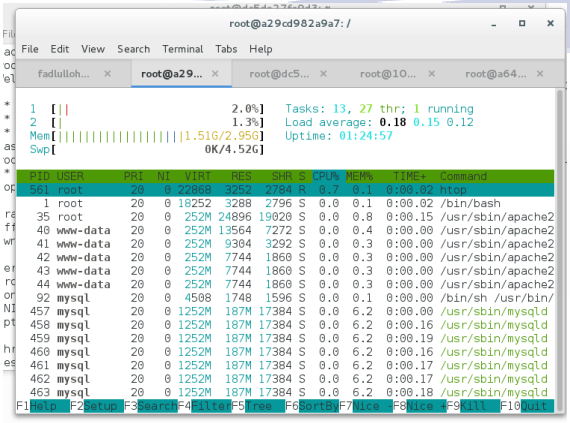
Gambar 7. *Docker inspect*

4. *Resources Memory*

Pada halaman dibawah ini merupakan halaman melihat kapasitas total memory dan besar memory yang sedang digunakan secara singkat dan sederhana.



Gambar 8. *host*



Gambar 9. *docker*

PENUTUP

Simpulan

Simpulan dari hasil kegiatan penelitian yang dilakukan adalah :

1. Telah berhasil dirancang dan diimplementasi *Lightweight Virtualization* dengan menggunakan Linux *Containers* (LXC) dan *Docker deployment* aplikasi *web* setelah dilakukan beberapa pengujian.
2. *Service web* pada masing-masing containers berjalan dengan baik sehingga semua aplikasi web dapat diakses oleh *client*.
3. Setiap aplikasi *web* beserta *environment* yang dibutuhkan berjalan pada lingkungan *virtual* (*virtual environment*) sehingga meminimalisir timbulnya masalah konflik dependensi.

Saran

1. Perlu adanya penelitian lebih lanjut dari sisi ekonomi mengenai adanya layanan semacam ini apakah dapat diaplikasikan dalam dunia nyata.
2. Penggunaan Docker untuk menjalankan proses *server Web* perlu diteliti lebih lanjut dalam hal konfigurasi yang cocok sehingga bisa menjalankan aplikasi *web* pengguna dengan efisien dan handal.

DAFTAR PUSTAKA

Adi Kurniawan, 2015, “Eksplorasi Pemanfaatan Docker untuk Mempermudah Pengelolaan Instalasi Komputer di Laboratorium Komputer Teknik Informatika Universitas Kristen Petra”, Surabaya.

Arif Rahman Hakim, 2015, “Desain dan Implementasi Lightweight Virtualization berbasis linux Containers dengan docker untuk deployment Aplikasi web”, Yogyakarta.

Docker, Installation Linux Debian. Website: https://docs.docker.com/engine/installation/linux/debian/#install-docker-ce, diakses tanggal 02 Feb 2017.

Firmansyah Adiputra,”Container dan Docker: Teknik Virtualisasi Dalam pengelolaan banyak Aplikasi web. Universitas Trunojoyo Madura, 2015

Masim vavai sugianto, Marsan susanto dkk. 2016**.** Virtualisasi Modern Berbasis Docker. PT Excellent Infotama Kreasindo, Bekasi Timur 17111.